How to Extract Text Between Characters in Google Sheets

Authored by

stats writer

February 18, 2026

The Power of Data Extraction in Modern Spreadsheets

In the contemporary landscape of digital productivity, **Google Sheets** has emerged as a cornerstone for professionals who require a versatile and robust platform for **data management**. The ability to manipulate large datasets within a **cloud-based** environment is not merely a convenience but a necessity for accurate reporting and business intelligence. One of the most sophisticated challenges users face is the precise isolation of **string** segments that are embedded within larger, more complex bodies of text. This process, often referred to as **data cleaning**, ensures that the information extracted for analysis is both relevant and free from extraneous noise, thereby improving the integrity of the resulting insights.

When working with unformatted data--such as logs, automated emails, or exported **metadata**--information is frequently concatenated into single cells, separated by specific delimiters or markers. To handle this effectively, users must go beyond basic cell editing and employ advanced **functions** that can interpret the structural patterns of the data. By mastering these techniques, a user can transform a chaotic **spreadsheet** into a streamlined source of truth, facilitating better decision-making processes across various organizational departments, from marketing to technical engineering.

This comprehensive guide focuses on the methodology of extracting text between two specific characters or strings using the most efficient tool available in the **Google Workspace** suite. While manual extraction might be feasible for a handful of entries, the scale of modern data requires an automated approach. By leveraging specialized formulas, users can ensure consistency across thousands of rows, significantly reducing the margin for human error and increasing the overall velocity of the data processing **workflow**. Understanding the underlying logic of these formulas is the first step toward becoming a proficient data analyst.

Ultimately, the objective is to empower the user to retrieve specific portions of text with surgical precision. Whether you are dealing with financial figures enclosed in brackets, product codes separated by hyphens, or narrative descriptions containing specific keywords, the principles of **pattern matching** remain the same. This article will provide a detailed technical walkthrough of the **REGEXEXTRACT** function, illustrating how it serves as the ultimate solution for complex text isolation tasks in **Google Sheets**.

Understanding the REGEXEXTRACT Functionality

The **REGEXEXTRACT** function is a powerful tool that utilizes **regular expressions**--a sequence of characters that defines a search pattern. Unlike simpler functions like FIND or SEARCH, which only look for static positions, REGEXEXTRACT interprets the logical structure of a string to identify and return specific matches. This makes it incredibly flexible for scenarios where the length of the

text or the exact position of the characters may vary from row to row. In the context of **data manipulation**, this flexibility is paramount for handling dynamic datasets.

To implement this function effectively, one must understand its **syntax** and how it interacts with the **Google Sheets** calculation engine. The function requires two primary arguments: the text to be searched (the cell reference) and the regular expression pattern itself (enclosed in quotation marks). When the formula is executed, the engine parses the cell content against the pattern and returns the first substring that matches the criteria defined within the **capture group**. This level of granularity allows for complex operations that would otherwise require nested formulas or custom scripts.

The beauty of using **regular expressions** within a spreadsheet environment is the standardization it brings to **automation**. Instead of writing unique logic for every possible variation of a text string, a single, well-crafted regex pattern can account for multiple edge cases simultaneously. This efficiency is why **REGEXEXTRACT** is considered an essential skill for anyone looking to optimize their use of **Google Sheets**. It bridges the gap between basic data entry and professional-grade data engineering.

Below is the fundamental syntax for isolating text between two specific boundaries, such as the words "this" and "that":

=REGEXEXTRACT(A2,"this(.*)that")

In this specific example, the formula is instructed to locate the word "this" and the word "that" within cell **A2** and extract everything that resides between them. The **wildcard** notation used here is crucial for the function's success, as it tells the system to capture any character of any length that exists between the two defined anchors. This approach is highly effective for cleaning up **log files** or extracting specific variables from a standardized text format.

Breaking Down the Regular Expression Syntax

To fully appreciate the utility of the **REGEXEXTRACT** formula, we must dissect the **regular expression** pattern used in the previous section. The pattern "this(.*)that" is composed of three distinct parts. The first and third parts ("this" and "that") act as anchors; they tell the function where to start and stop looking. The middle part, `(.*)`, is the **capture group**. The parentheses indicate that the text matched by the symbols inside should be the output of the function, rather than the entire match including the anchors.

Inside the capture group, the period `.` is a **metacharacter** that represents any single character, including letters, numbers, spaces, and punctuation. The asterisk `*` is a **quantifier** that instructs the engine to match the preceding character (in this case, the period) zero or more times.

Together, `.*` acts as a "greedy" operator, capturing as much text as possible between the defined boundaries. This is fundamental for **information extraction** tasks where the length of the internal string is unknown or varies between rows.

Understanding these symbols allows users to build more complex queries. For instance, if you only wanted to match digits between two characters, you could replace the period with a digit class. However, for most general **data cleaning** tasks in **Google Sheets**, the `(.*)` pattern provides the perfect balance of simplicity and power. It treats the data as a raw **character stream**, ensuring that nothing is missed during the extraction process.

It is also important to note that regular expressions are case-sensitive by default in **Google Sheets**. If your data contains "This" with a capital letter but your formula uses "this," the match will fail. Mastery of **syntax** involves not only knowing the symbols but also being aware of the literal content of your data. This attention to detail is what distinguishes an expert user from a novice when performing high-level **data manipulation**.

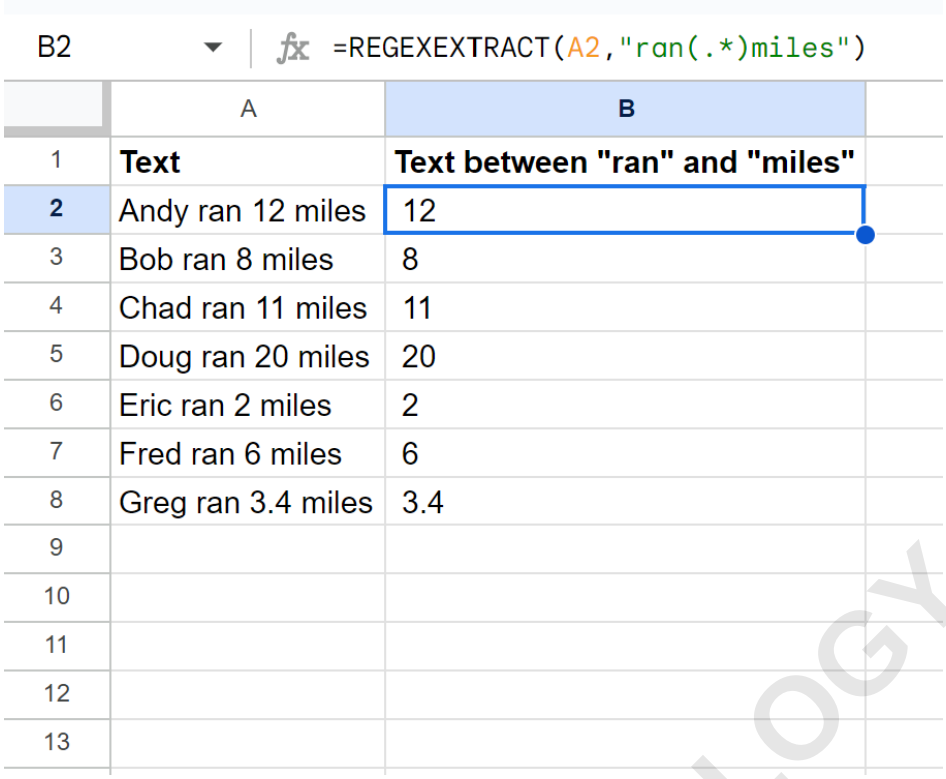
Practical Application: Extracting Content Between Text Strings

Let us examine a real-world scenario where a user needs to process activity logs. Imagine a column where each cell contains a sentence describing a workout, such as "John ran 5.2 miles yesterday." If the goal is to extract only the numerical value (the distance), we can identify the words "ran" and "miles" as our anchors. By applying the **REGEXEXTRACT** function, we can isolate the content between these two strings efficiently across the entire **spreadsheet**.

To execute this, you would enter the following formula into cell **B2** to analyze the data in cell **A2**:

=REGEXEXTRACT(A2,"ran(.*)miles")

Once the formula is applied to the first cell, the **Google Sheets** "fill handle" (the small square at the bottom-right of the cell) can be used to drag the formula down the entire column. This action automatically updates the cell references, applying the same extraction logic to every row. This is a prime example of how **automation** can save hours of manual labor, especially when dealing with hundreds or thousands of records.



	A	B
1	**Text**	**Text between "ran" and "miles"**
2	Andy ran 12 miles	12
3	Bob ran 8 miles	8
4	Chad ran 11 miles	11
5	Doug ran 20 miles	20
6	Eric ran 2 miles	2
7	Fred ran 6 miles	6
8	Greg ran 3.4 miles	3.4
9		
10		
11		
12		
13		

The resulting values in Column B will strictly contain the text that was sandwiched between "ran" and "miles." This extracted data is now in a clean, isolated format, making it ready for mathematical operations, **data visualization**, or further **statistical analysis**. By separating the numeric data from the narrative text, the user has transformed raw information into actionable data points.

This method is particularly useful when the text surrounding the target data is consistent, but the target data itself is variable. In our exercise, whether the person ran 5 miles or 12.5 miles, the formula remains valid because the `(.*)` **wildcard** adapts to the length of the string. This demonstrates the robustness of **regular expressions** in handling real-world data variability within **Google Sheets**.

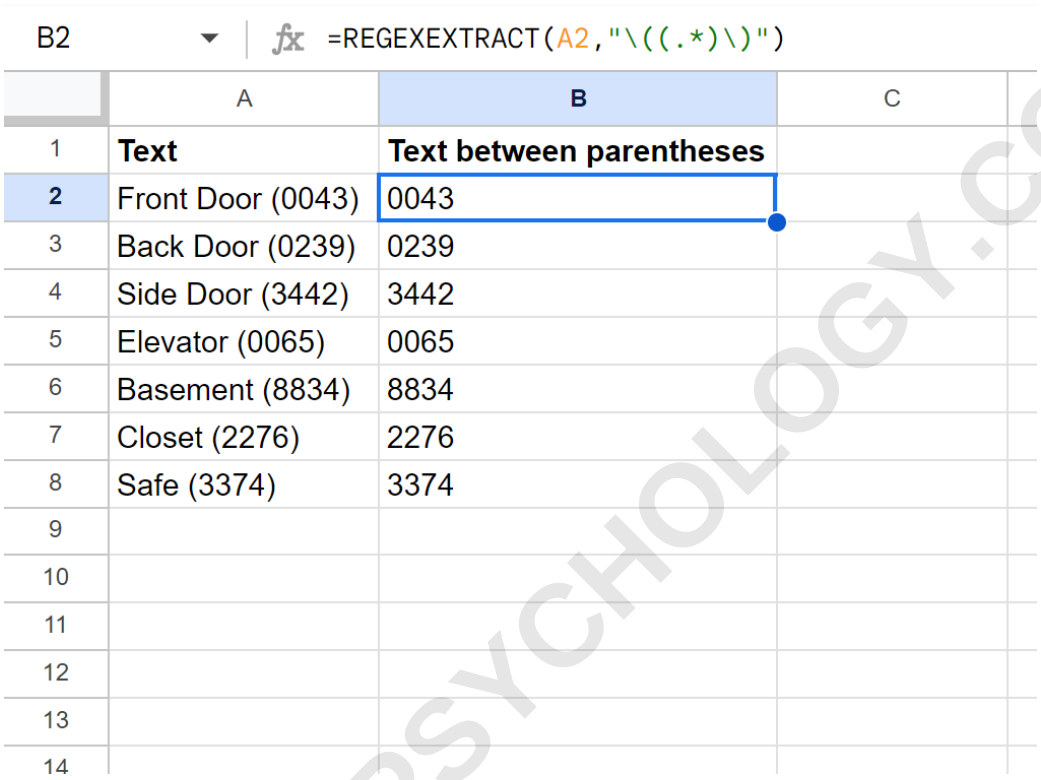
Advanced Use Case: Isolating Text Within Parentheses

Another frequent requirement in **data cleaning** is extracting information that has been placed inside parentheses. This is common in datasets involving names with IDs (e.g., "John Doe (ID1234)"), or financial reports where secondary information is bracketed. However, because parentheses are **metacharacters** in regex--used to define capture groups--they require special handling to be treated as literal characters to be matched.

To extract text between parentheses in cell **A2**, we use the following specific **syntax** in cell **B2**:

=REGEXEXTRACT(A2,"((.*))")

In this formula, the backslash `` serves as an **escape character**. It tells the regex engine to treat the subsequent parenthesis as a literal character to look for in the text, rather than as a functional part of the regex command. The inner parentheses--the ones not preceded by a backslash--form the actual **capture group** that identifies which part of the match to return as the result.



	A	B	C
1	**Text**	**Text between parentheses**	
2	Front Door (0043)	0043	
3	Back Door (0239)	0239	
4	Side Door (3442)	3442	
5	Elevator (0065)	0065	
6	Basement (8834)	8834	
7	Closet (2276)	2276	
8	Safe (3374)	3374	
9			
10			
11			
12			
13			
14			

Applying this logic across a dataset allows for the rapid isolation of supplemental information. As shown in the visual example, Column B successfully captures the content previously held within brackets in Column A. This technique is essential for anyone working with **databases** or **API** exports where parentheses are a standard convention for containing specific identifiers or **attributes**.

Without the **escape character**, the formula would result in a **parsing** error or incorrect results, as the engine would be confused by the nested parentheses. This highlights the importance of understanding the technical nuances of **regular expressions**. When you master the use of escape characters, you unlock the ability to target almost any symbol or punctuation mark within a **Google Sheets** cell.

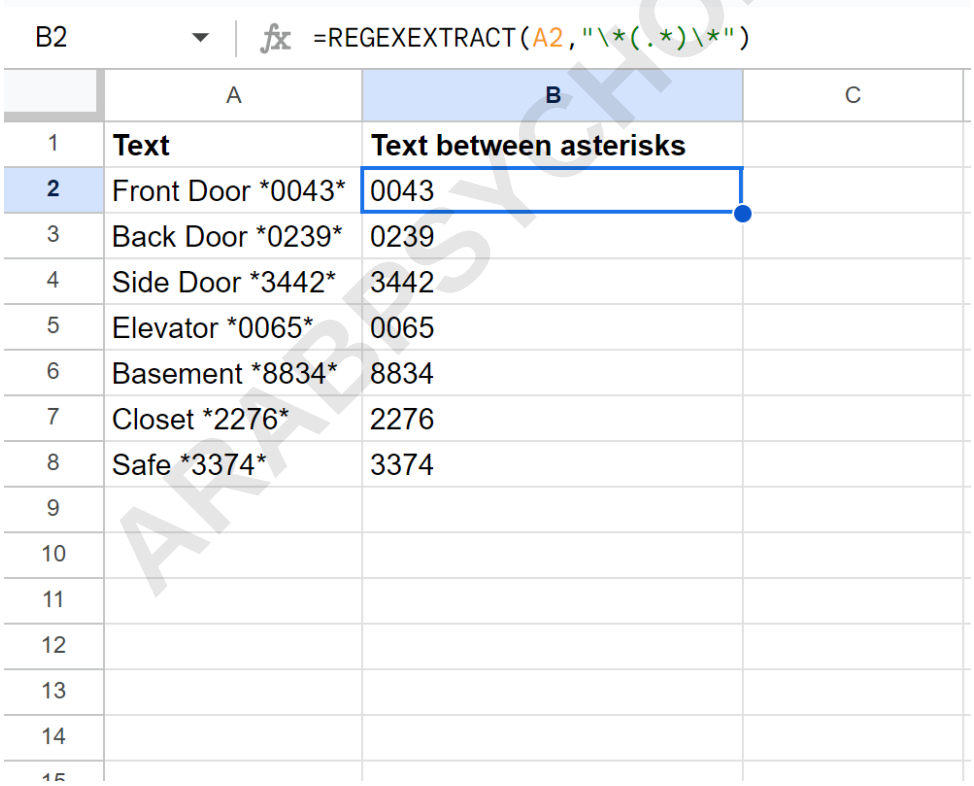
Navigating Special Characters and Escape Sequences

Special characters, such as asterisks, plus signs, and question marks, hold specific meanings within the **regular expression** language. If you need to extract text between two asterisks--a common format in markdown or specific configuration files--you must once again utilize the **escape character**. Failing to do so would lead the **REGEXEXTRACT** function to interpret the asterisk as a **quantifier**, which would break the logic of your search pattern.

To extract text between two asterisks in cell **A2**, use this formula in cell **B2**:

=REGEXEXTRACT(A2,"*(.*)*")

By placing the backslash before each asterisk, you ensure that the formula specifically looks for the literal "*" character. This allows the `(.*)` **wildcard** to accurately capture the string contained between them. This process is vital for **information retrieval** in environments where symbols are used as delimiters for important data points, such as bolded text in a raw export or specific tags in a **markup language**.



	A	B	C
1	**Text**	**Text between asterisks**	
2	Front Door *0043*	0043	
3	Back Door *0239*	0239	
4	Side Door *3442*	3442	
5	Elevator *0065*	0065	
6	Basement *8834*	8834	
7	Closet *2276*	2276	
8	Safe *3374*	3374	
9			
10			
11			
12			
13			
14			

As illustrated, Column B now contains only the text that was originally surrounded by asterisks in Column A. This level of control is what makes **Google Sheets** such a powerful tool for **data processing**. By learning which characters are "special" and require escaping, you can build

incredibly resilient formulas that handle even the most complex **string** patterns without failing or producing errors.

It is worth noting that if your dataset contains multiple instances of the same character, the "greedy" nature of the `*` quantifier might capture more than you intend. In more advanced **regular expression** applications, you might use a "lazy" quantifier (by adding a question mark, like `.*?`) to stop at the first possible closing character. However, for most standard tasks in **Google Sheets**, the basic escape and capture pattern is more than sufficient for high-quality **data cleaning**.

The Strategic Importance of Precise Data Cleaning

Effective **data cleaning** is the foundation of any successful **data analysis** project. When data is cluttered with unnecessary characters, strings, or formatting, it becomes difficult to perform **aggregations** or comparisons. By using tools like **REGEXEXTRACT** to isolate specific text, you ensure that your dataset is standardized. Standardization is key to ensuring that formulas like SUMIF, VLOOKUP, or pivot tables function correctly without being tripped up by hidden spaces or extra characters.

Furthermore, precise extraction is essential for **data integrity**. In a professional setting, relying on manual "copy and paste" is a recipe for disaster, as it introduces human fatigue and inconsistency. Automating the extraction of text between characters ensures that the same logic is applied to every single row, regardless of whether you have ten rows or ten thousand. This scalability is a hallmark of professional **spreadsheet** management and a vital skill for anyone working in **data science** or administrative roles.

Beyond simple extraction, these techniques allow for the creation of new **dimensions** within your data. For example, by extracting a city name from a full address string, you can now perform geographical analysis that was previously impossible. By isolating a product code from a description, you can link your spreadsheet to an external **inventory management system**. In this way, text extraction is not just about cleaning; it is about enriching your data and making it more useful for the organization.

Finally, mastering these **Google Sheets** functions contributes to a more efficient **business process**. Time spent manually fixing data is time lost for analysis and strategy. By implementing robust **regular expression** patterns, you create self-maintaining systems where new data entries are automatically processed and cleaned. This move toward **automation** is a significant step in optimizing any data-driven workflow.

Comparative Analysis: REGEXEXTRACT vs. Traditional Functions

While **REGEXEXTRACT** is incredibly powerful, it is helpful to understand how it compares to

traditional **Google Sheets** functions like MID, FIND, and SEARCH. For very simple tasks, such as extracting the first five characters of a string, the LEFT function is perfectly adequate. However, when the position of the target text changes--for instance, if the word you want is sometimes at the beginning and sometimes in the middle--traditional functions often require complex, "nested" logic that is difficult to read and maintain.

A nested formula using MID and FIND to extract text between two characters might look like this: `=MID(A2, FIND("this", A2) + 4, FIND("that", A2) - FIND("this", A2) - 4)`. While this works, it is prone to errors if one of the strings is missing and is generally much harder to debug than a **regular expression**. REGEXEXTRACT replaces this convoluted logic with a single, elegant pattern, making your **spreadsheet** cleaner and more professional.

Another advantage of **regular expressions** is their ability to handle **pattern matching** rather than just literal matching. If you need to extract text between "any number" and "any capital letter," REGEXEXTRACT can do this easily with character classes, whereas traditional functions would fail completely. This makes REGEXEXTRACT the superior choice for any **data manipulation** task that involves structural complexity or unpredictable data formats.

However, it is worth noting that **REGEXEXTRACT** can be more computationally intensive than basic text functions. In extremely large spreadsheets with hundreds of thousands of regex formulas, you might notice a slight impact on performance. Nonetheless, for the vast majority of users, the benefits of accuracy, flexibility, and readability far outweigh any minor performance considerations. Choosing the right tool for the job is a key part of **software engineering** principles applied to spreadsheets.

Expanding Your Google Sheets Automation Skills

Mastering the extraction of text between two characters is just the beginning of what you can achieve with **regular expressions** in **Google Sheets**. Once you are comfortable with **REGEXEXTRACT**, you can explore related functions like REGEXREPLACE and REGEXMATCH. REGEXREPLACE allows you to swap out parts of a string based on a pattern, which is perfect for reformatting dates or phone numbers. REGEXMATCH returns a simple TRUE or FALSE, which is excellent for **data validation** or filtering lists.

To further enhance your **automation** capabilities, consider combining these functions with other Google Sheets features. For instance, using REGEXEXTRACT inside an ARRAYFORMULA allows you to process an entire column with a single formula in the top cell. This not only makes your sheet faster but also prevents the need to manually drag formulas down when new data is added. This is a common practice in **DevOps** and data engineering to ensure systems are as hands-off as possible.

If you find yourself hitting the limits of built-in functions, you might also explore **Google Apps Script**. This **JavaScript**-based platform allows you to write custom functions and automate complex workflows across the entire **Google Workspace**. However, for 90% of text manipulation tasks, a solid understanding of **regular expressions** within the standard grid will be more than sufficient to handle your needs.

The journey to becoming a **power user** involves continuous learning and practice. We encourage you to experiment with different patterns and explore the official documentation to see the full range of possibilities. To continue your education, check out the following tutorials that explain how to perform other common tasks in Google Sheets:

Google Sheets: How to Extract Text After a Character
Google Sheets: How to Extract Text Before a Character

By integrating these techniques into your daily routine, you will significantly improve your **data processing** efficiency and accuracy. Whether you are managing personal projects or enterprise-level datasets, the ability to manipulate text with **regular expressions** is a transformative skill that will serve you well throughout your career in the digital age.